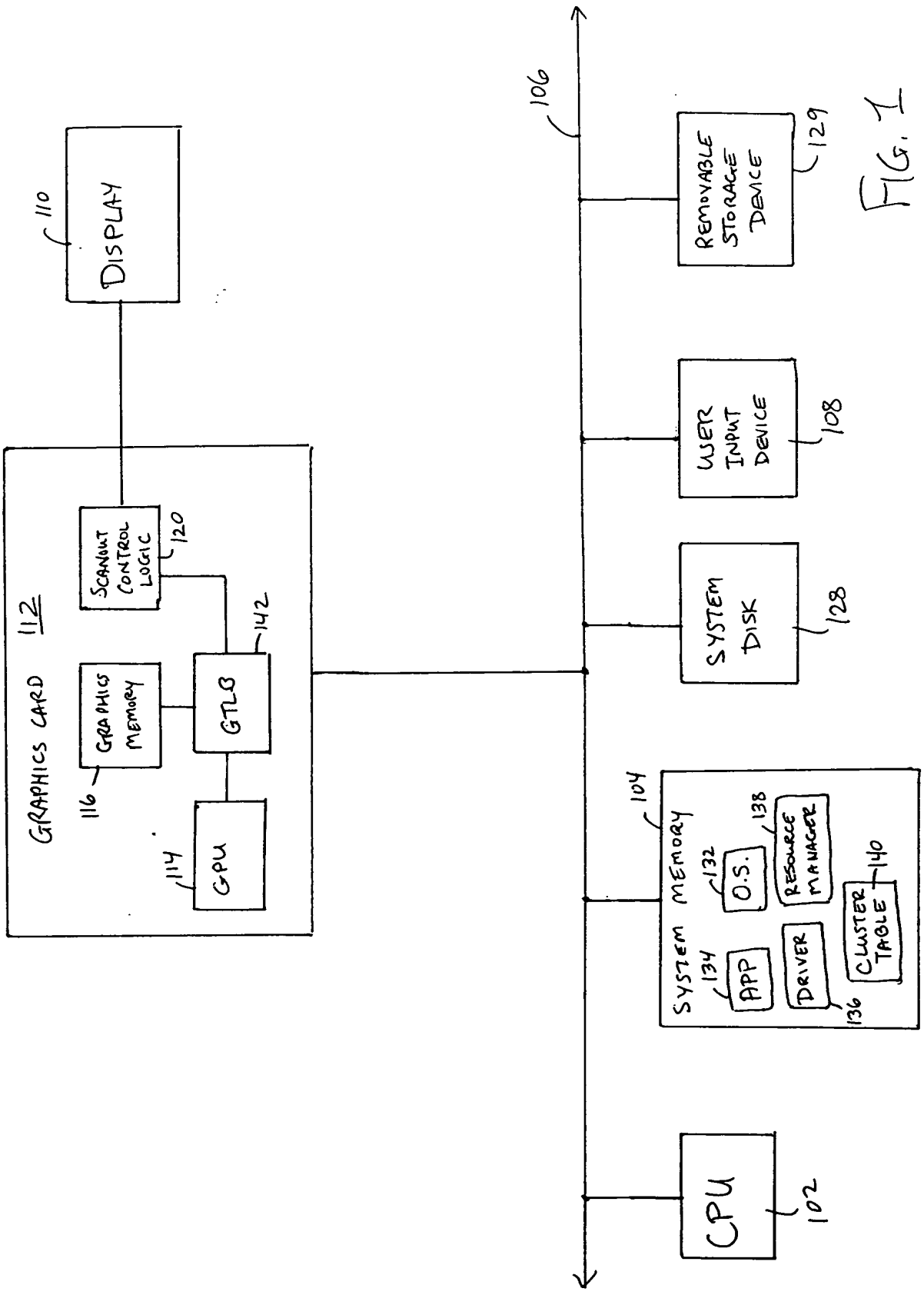
106
DISPLAY
110
REMOVABLE STORAGE DEVICE
129
USER INPUT DEVICE
108
SYSTEM DISK
128
GRAPHICS CARD
112
SCANOUT CONTROL LOGIC
120
GRAPHICS MEMORY
116
GTLB
142
GPU
114
SYSTEM MEMORY
104
O.S.
132
138
RESOURCE MANAGER
APP
134
DRIVER
136
CLUSTER TABLE
140
CPU
102

FIG. 1

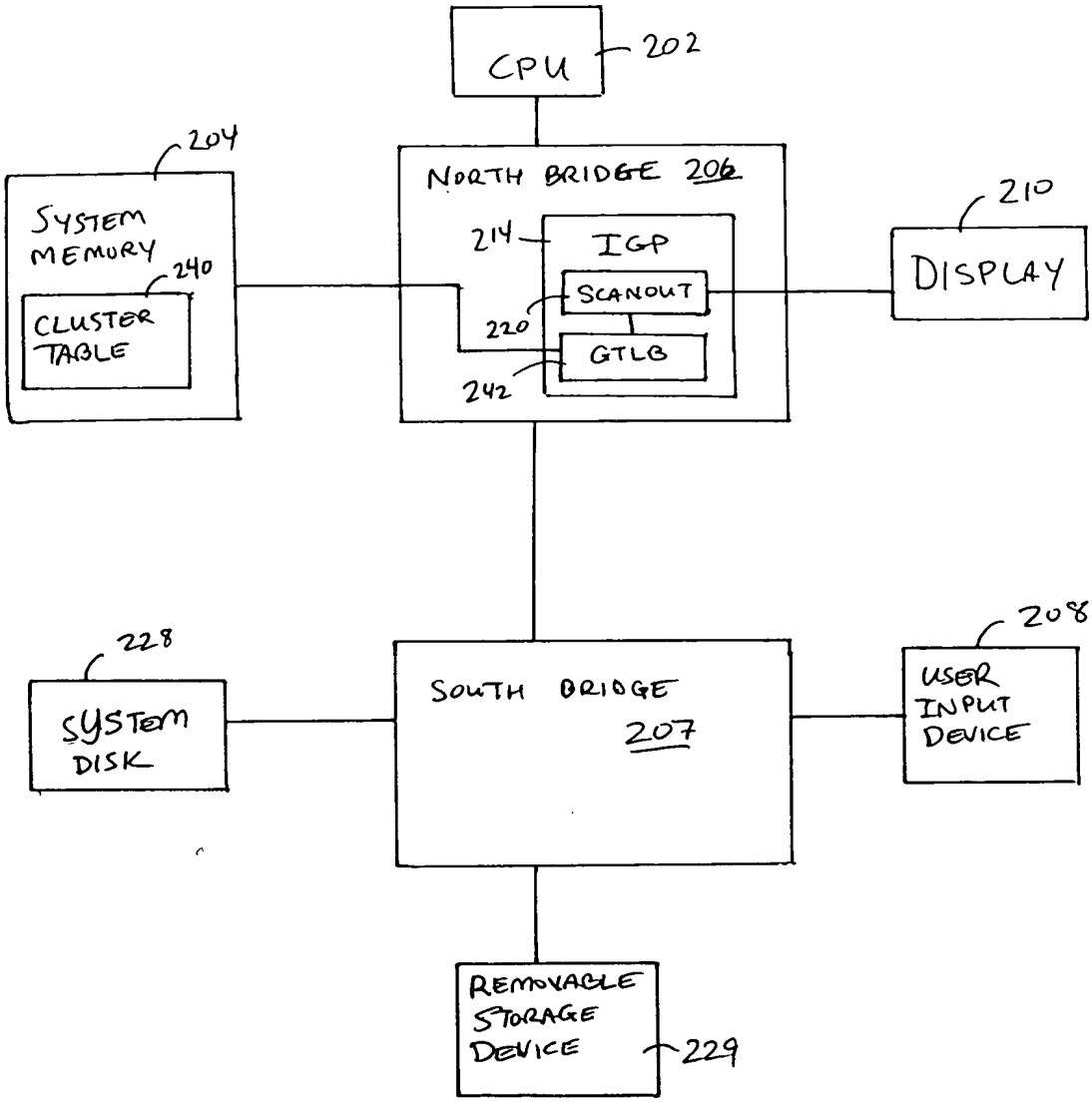
CPU
202
NORTH BRIDGE 206
204
SYSTEM MEMORY
240
CLUSTER TABLE
214
IGP
SCANOUT
220
GTLB
242
210
DISPLAY
228
SYSTEM DISK
SOUTH BRIDGE 207
208
USER INPUT DEVICE
REMOVABLE STORAGE DEVICE
229

FIG. 2

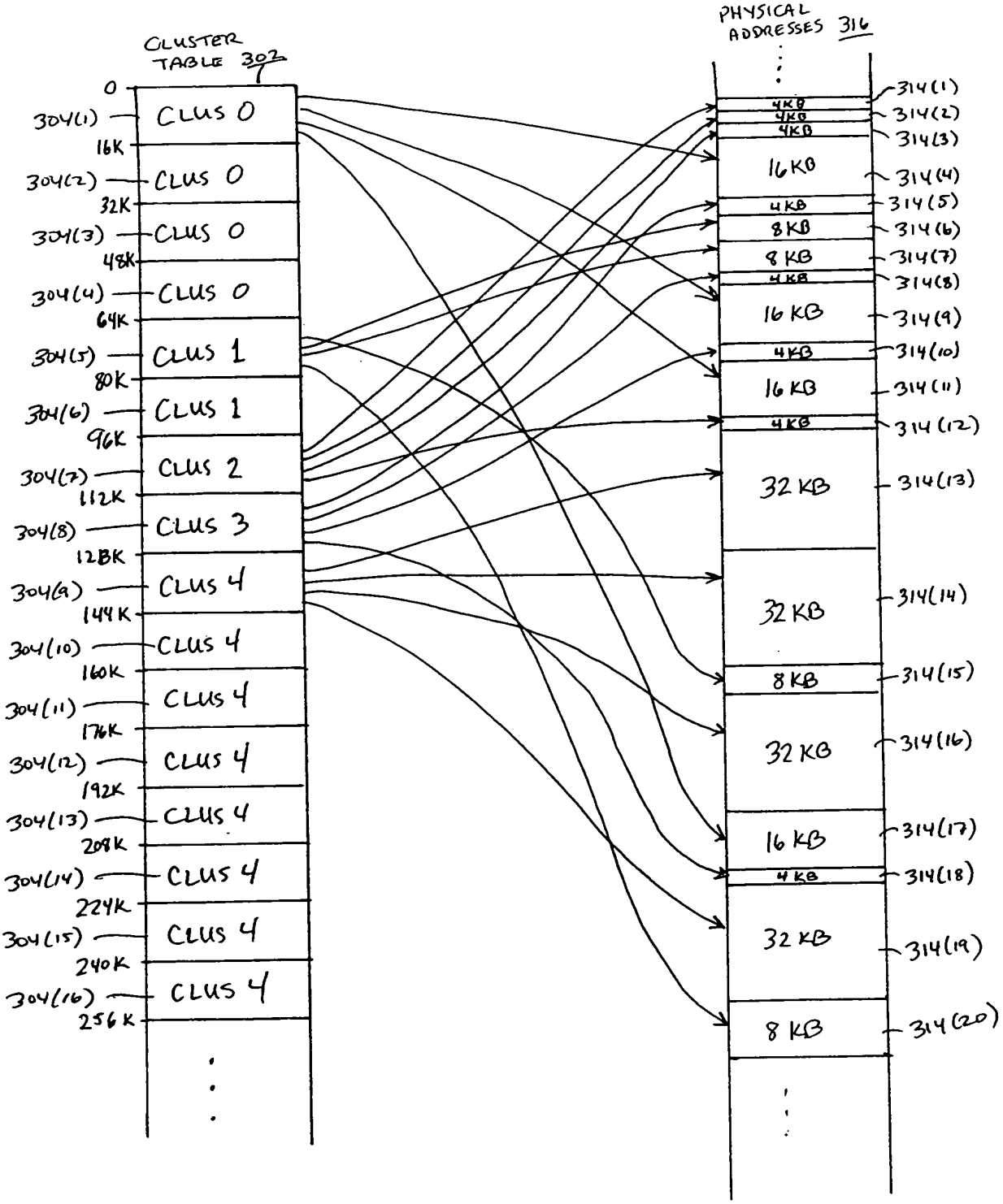
CLUSTER TABLE 302
PHYSICAL ADDRESSES 316
0
304(1)
CLUS 0
16K
304(2)
CLUS 0
32K
304(3)
CLUS 0
48K
304(4)
CLUS 0
64K
304(5)
CLUS 1
80K
304(6)
CLUS 1
96K
304(7)
CLUS 2
112K
304(8)
CLUS 3
128K
304(9)
CLUS 4
144K
304(10)
CLUS 4
160K
304(11)
CLUS 4
176K
304(12)
CLUS 4
192K
304(13)
CLUS 4
208K
304(14)
CLUS 4
224K
304(15)
CLUS 4
240K
304(16)
CLUS 4
256K
4KB
4KB
4KB
16 KB
4KB
8 KB
8 KB
4KB
16 KB
4KB
16 KB
4KB
32 KB
32 KB
8 KB
32 KB
16 KB
4KB
32 KB
8 KB
314(1)
314(2)
314(3)
314(4)
314(5)
314(6)
314(7)
314(8)
314(9)
314(10)
314(11)
314(12)
314(13)
314(14)
314(15)
314(16)
314(17)
314(18)
314(19)
314(20)

FIG. 3

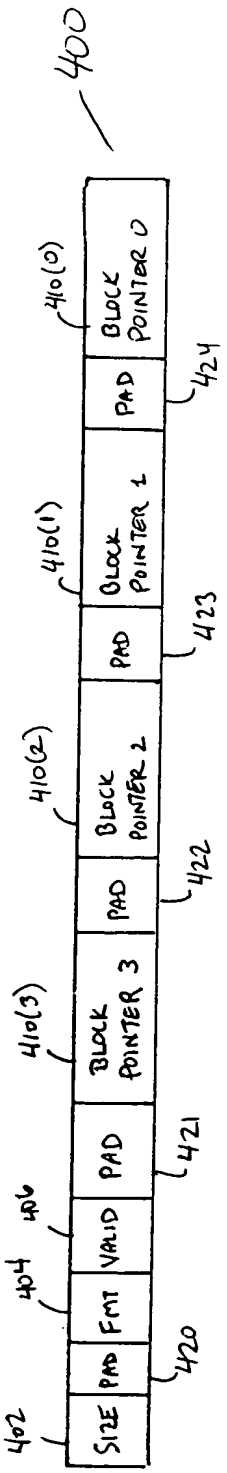
400
410(0)
BLOCK POINTER 0
PAD
424
410(1)
BLOCK POINTER 1
PAD
423
410(2)
BLOCK POINTER 2
PAD
422
410(3)
BLOCK POINTER 3
PAD
421
406
VALID
404
FMT
PAD
420
402
SIZE

FIG. 4A

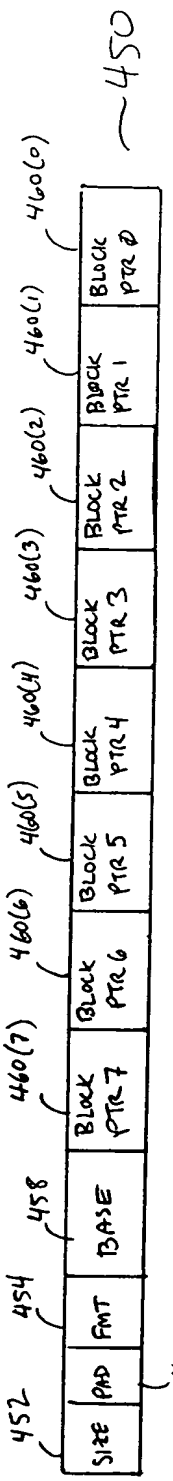
450
460(0)
BLOCK PTR 0
460(1)
BLOCK PTR 1
460(2)
BLOCK PTR 2
460(3)
BLOCK PTR 3
460(4)
BLOCK PTR 4
460(5)
BLOCK PTR 5
460(6)
BLOCK PTR 6
460(7)
BLOCK PTR 7
458
BASE
454
FMT
PAD
470
452
SIZE

FIG. 4B

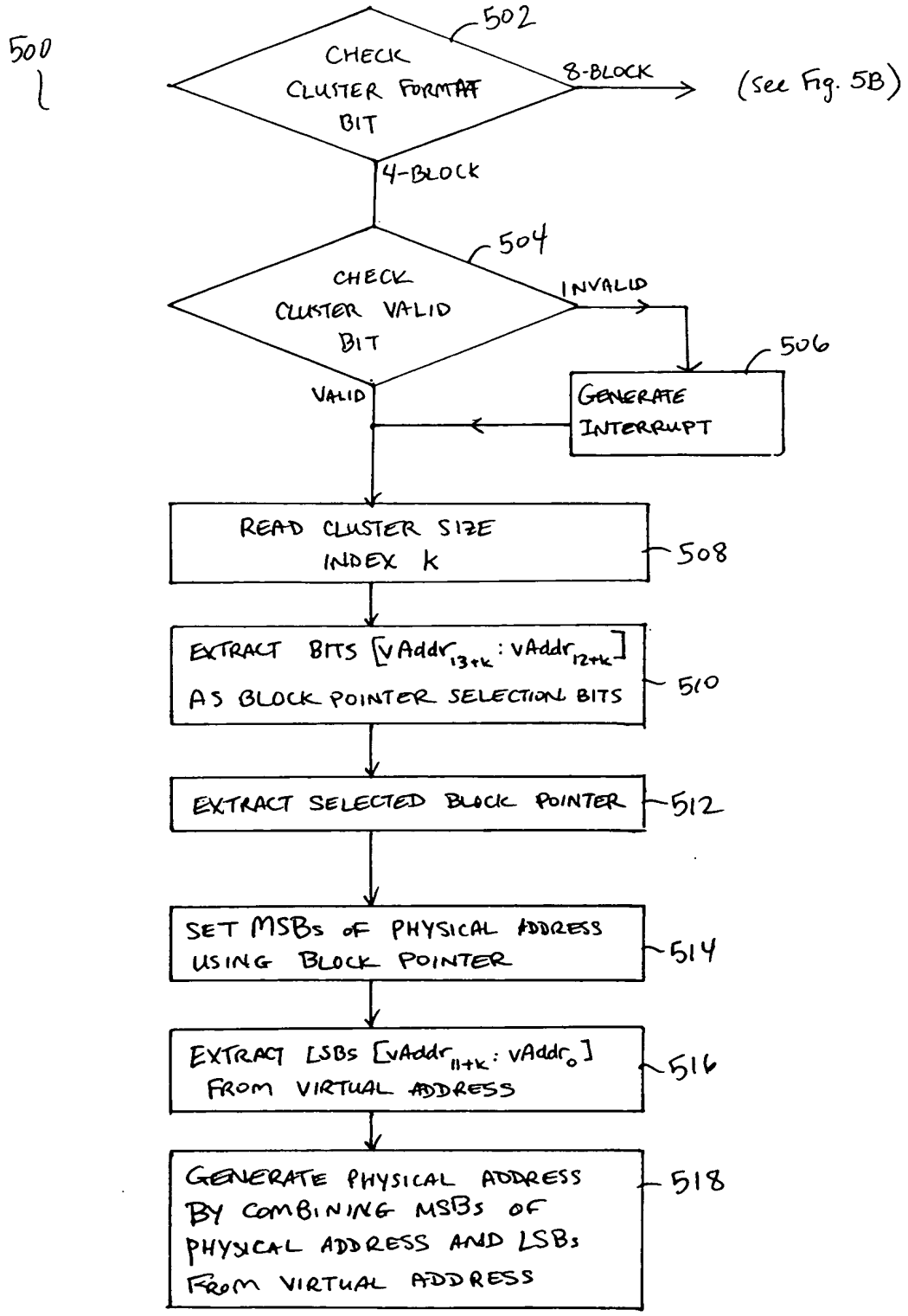
500
502
CHECK CLUSTER FORMAT BIT
8-BLOCK
(see Fig. 5B)
4-BLOCK
504
CHECK CLUSTER VALID BIT
INVALID
506
GENERATE INTERRUPT
VALID
READ CLUSTER SIZE INDEX k
508
EXTRACT BITS [vAddr13+k : vAddr12+k] AS BLOCK POINTER SELECTION BITS
510
EXTRACT SELECTED BLOCK POINTER
512
SET MSBs OF PHYSICAL ADDRESS USING BLOCK POINTER
514
EXTRACT LSBs [vAddr11+k : vAddr0] FROM VIRTUAL ADDRESS
516
GENERATE PHYSICAL ADDRESS BY COMBINING MSBs OF PHYSICAL ADDRESS AND LSBs FROM VIRTUAL ADDRESS
518

FIG. 5A

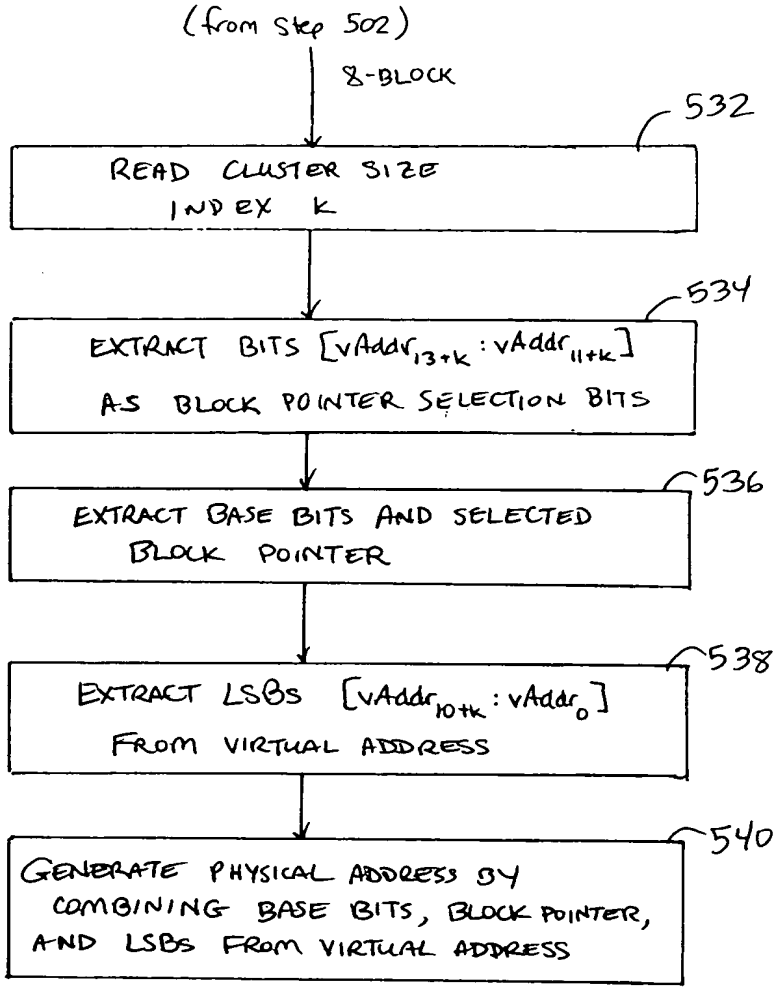
(from Step 502)
8-BLOCK
532
READ CLUSTER SIZE INDEX k
534
EXTRACT BITS [vAddr13+k : vAddr11+k] AS BLOCK POINTER SELECTION BITS
536
EXTRACT BASE BITS AND SELECTED BLOCK POINTER
538
EXTRACT LSBs [vAddr10+k : vAddr0] FROM VIRTUAL ADDRESS
540
GENERATE PHYSICAL ADDRESS BY COMBINING BASE BITS, BLOCK POINTER, AND LSBs FROM VIRTUAL ADDRESS

FIG. 5B

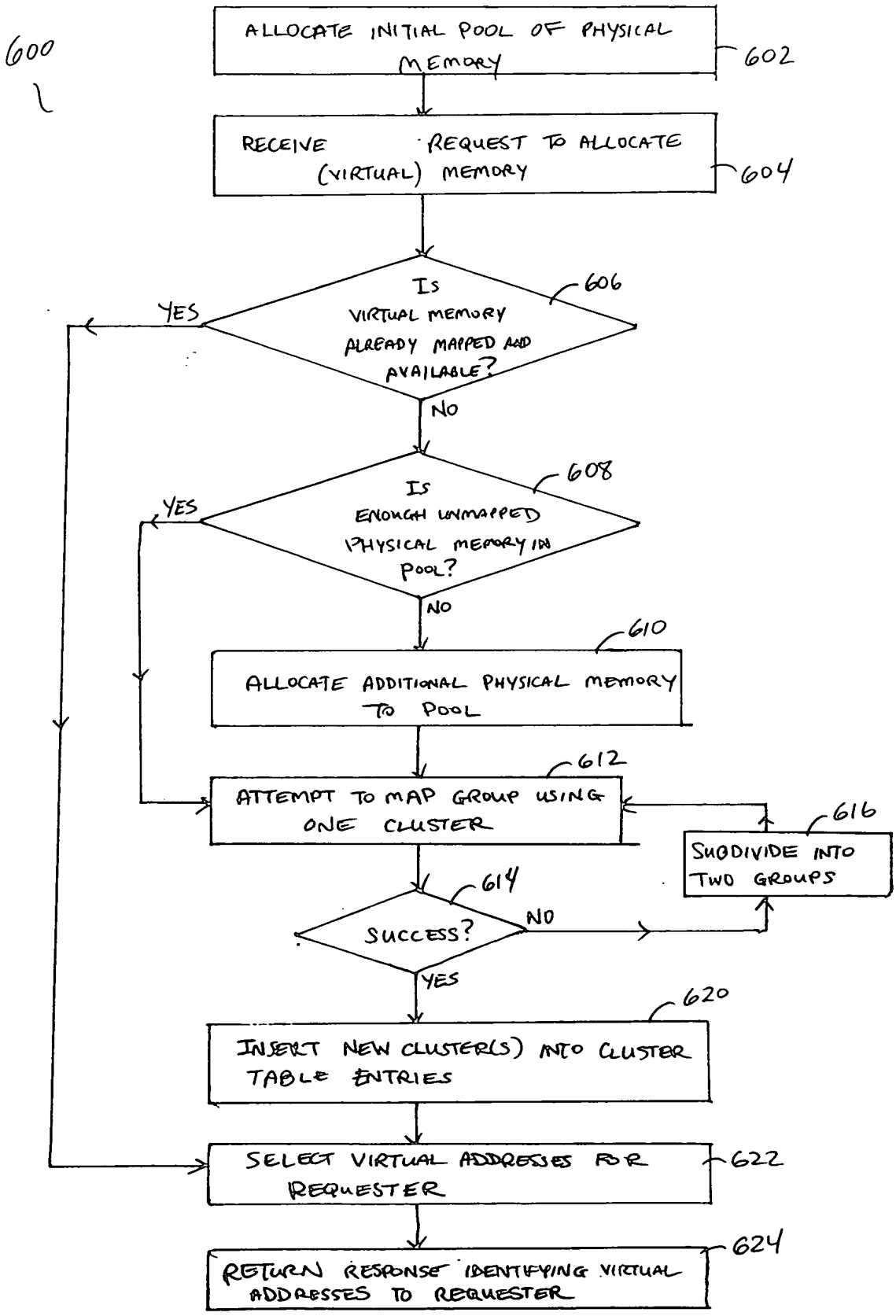
600
ALLOCATE INITIAL POOL OF PHYSICAL MEMORY
602
RECEIVE REQUEST TO ALLOCATE (VIRTUAL) MEMORY
604
IS VIRTUAL MEMORY ALREADY MAPPED AND AVAILABLE?
606
YES
NO
IS ENOUGH UNMAPPED PHYSICAL MEMORY IN POOL?
608
YES
NO
ALLOCATE ADDITIONAL PHYSICAL MEMORY TO POOL
610
ATTEMPT TO MAP GROUP USING ONE CLUSTER
612
SUBDIVIDE INTO TWO GROUPS
616
SUCCESS?
614
NO
YES
INSERT NEW CLUSTER(S) INTO CLUSTER TABLE ENTRIES
620
SELECT VIRTUAL ADDRESSES FOR REQUESTER
622
RETURN RESPONSE IDENTIFYING VIRTUAL ADDRESSES TO REQUESTER
624

FIG. 6

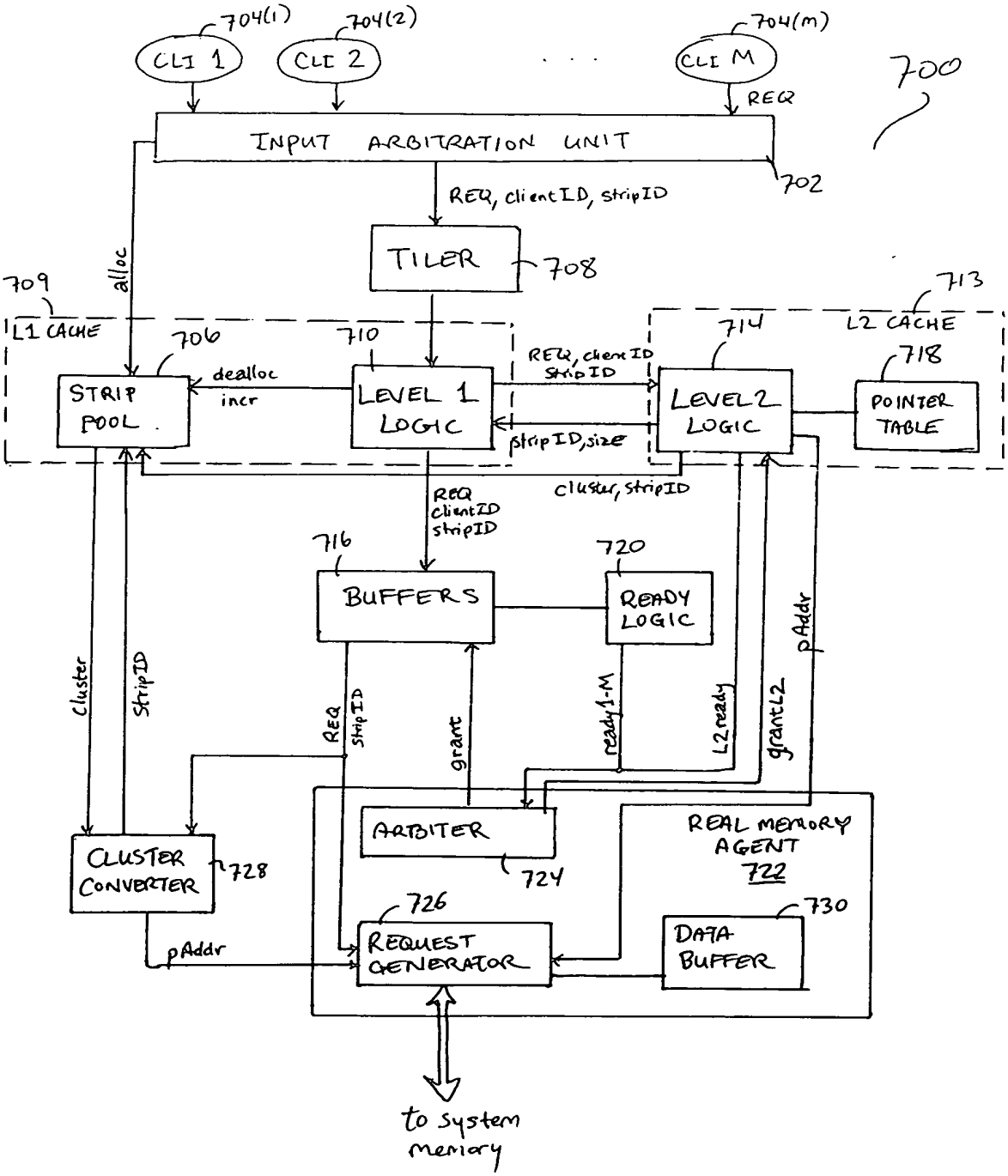
704(1)
704(2)
704(m)
CLI 1
CLI 2
CLI M
REQ
700
INPUT ARBITRATION UNIT
702
REQ, clientID, stripID
TILER
708
alloc
709
L1 CACHE
713
L2 CACHE
706
710
714
718
STRIP POOL
dealloc
incr
LEVEL 1 LOGIC
REQ, clientID stripID
stripID, size
LEVEL 2 LOGIC
POINTER TABLE
cluster, stripID
REQ clientID stripID
716
720
BUFFERS
READY LOGIC
pAddr
Cluster
StripID
REQ
stripID
grant
ready 1-M
L2 ready
grant L2
ARBITER
724
REAL MEMORY AGENT
722
CLUSTER CONVERTER
728
726
730
REQUEST GENERATOR
DATA BUFFER
pAddr
to system memory

FIG. 7

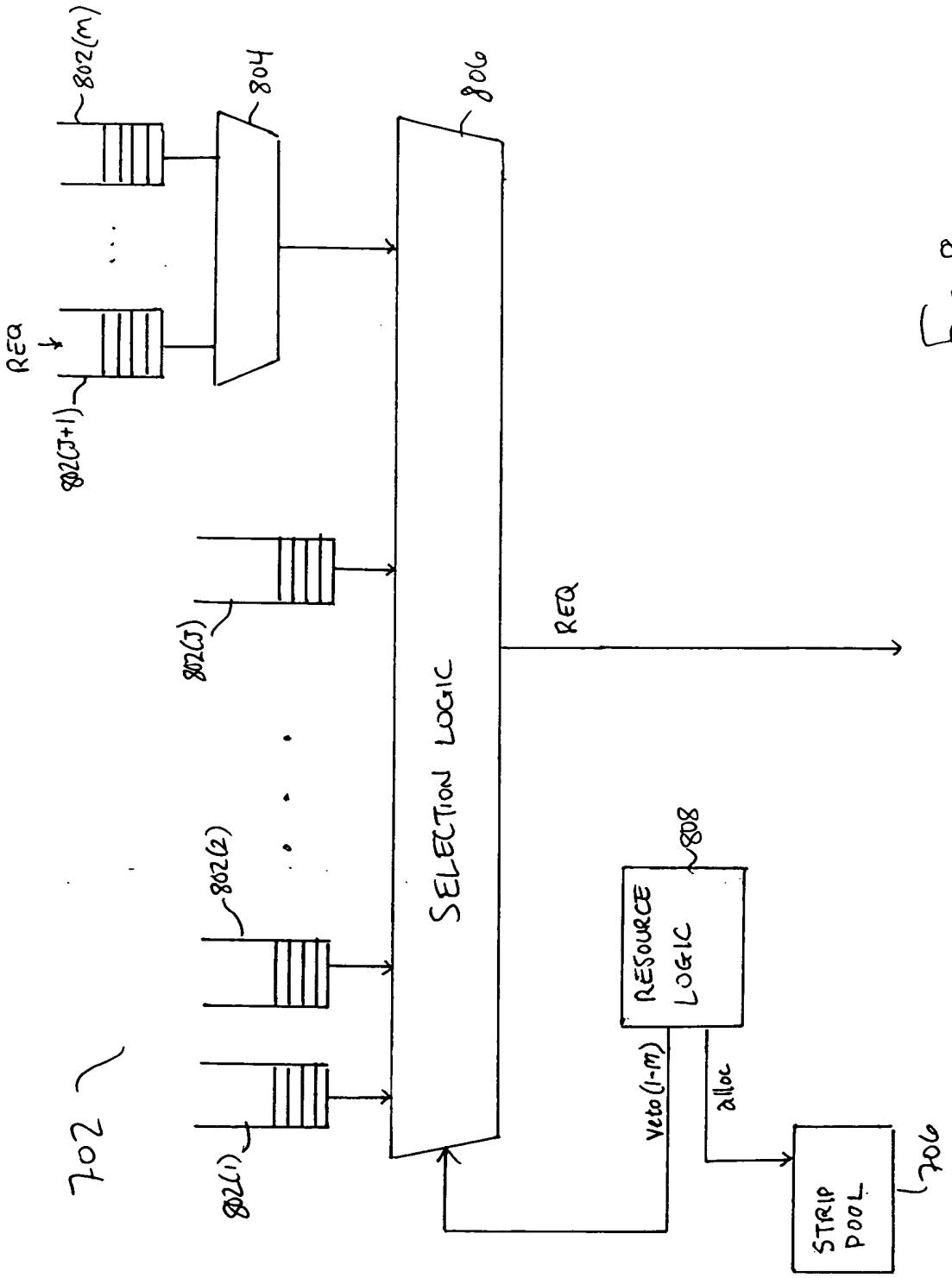
802(m)
804
806
REQ
802(J+1)
802(J)
REQ
SELECTION LOGIC
802(2)
808
RESOURCE LOGIC
802(1)
702
Veto(1-m)
alloc
STRIP POOL
706

Fig. 8

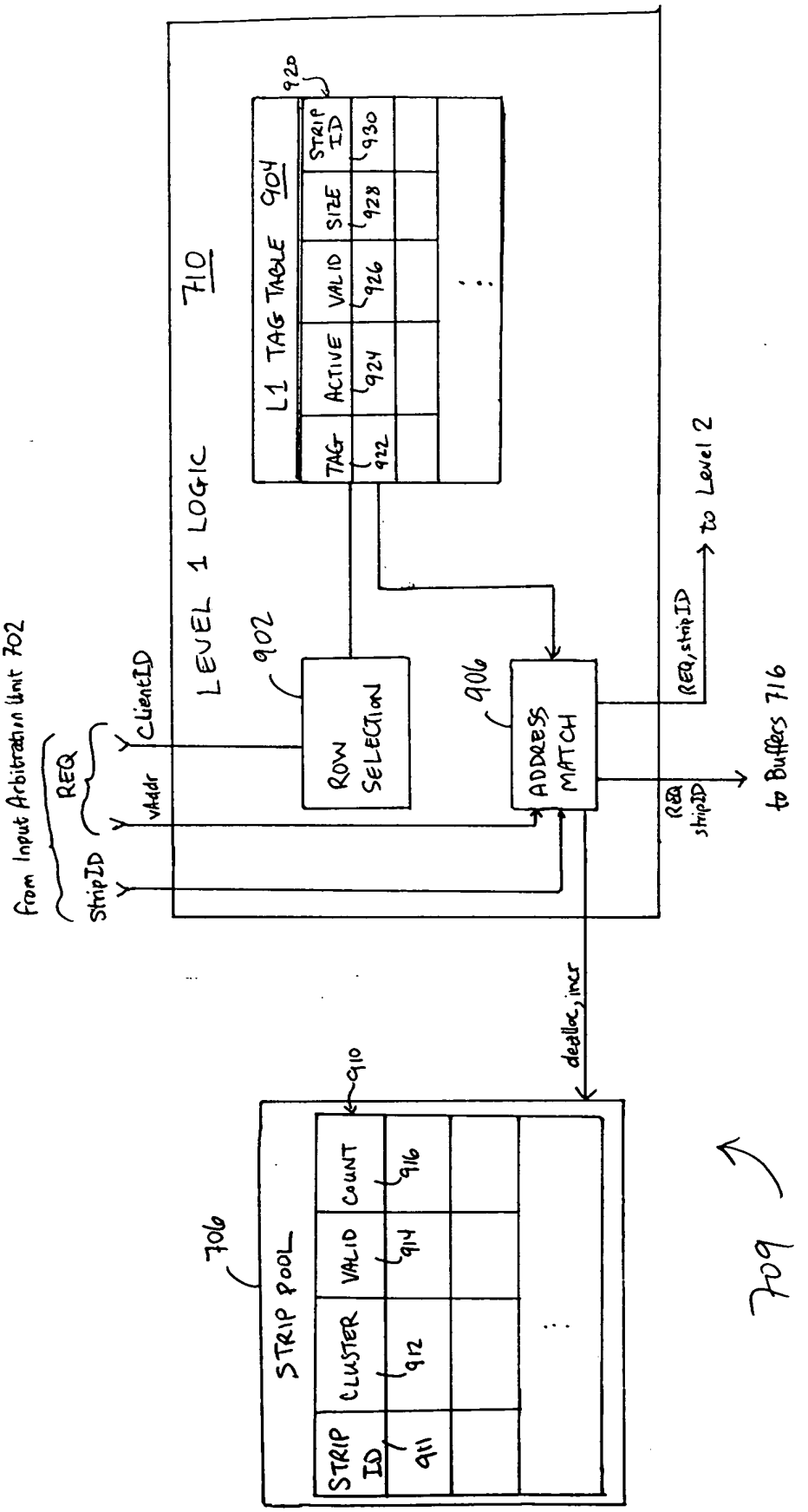
710
LEVEL 1 LOGIC
L1 TAG TABLE 904
920
TAG
ACTIVE
VALID
SIZE
STRIP ID
922
924
926
928
930
902
ROW SELECTION
906
ADDRESS MATCH
From Input Arbitration Unit 702
REQ
ClientID
vAddr
StripID
REQ, stripID
to Level 2
REQ stripID
to Buffers 716
deallocs, incr
706
STRIP POOL
910
STRIP ID
CLUSTER
VALID
COUNT
911
912
914
916
709

FIG. 9

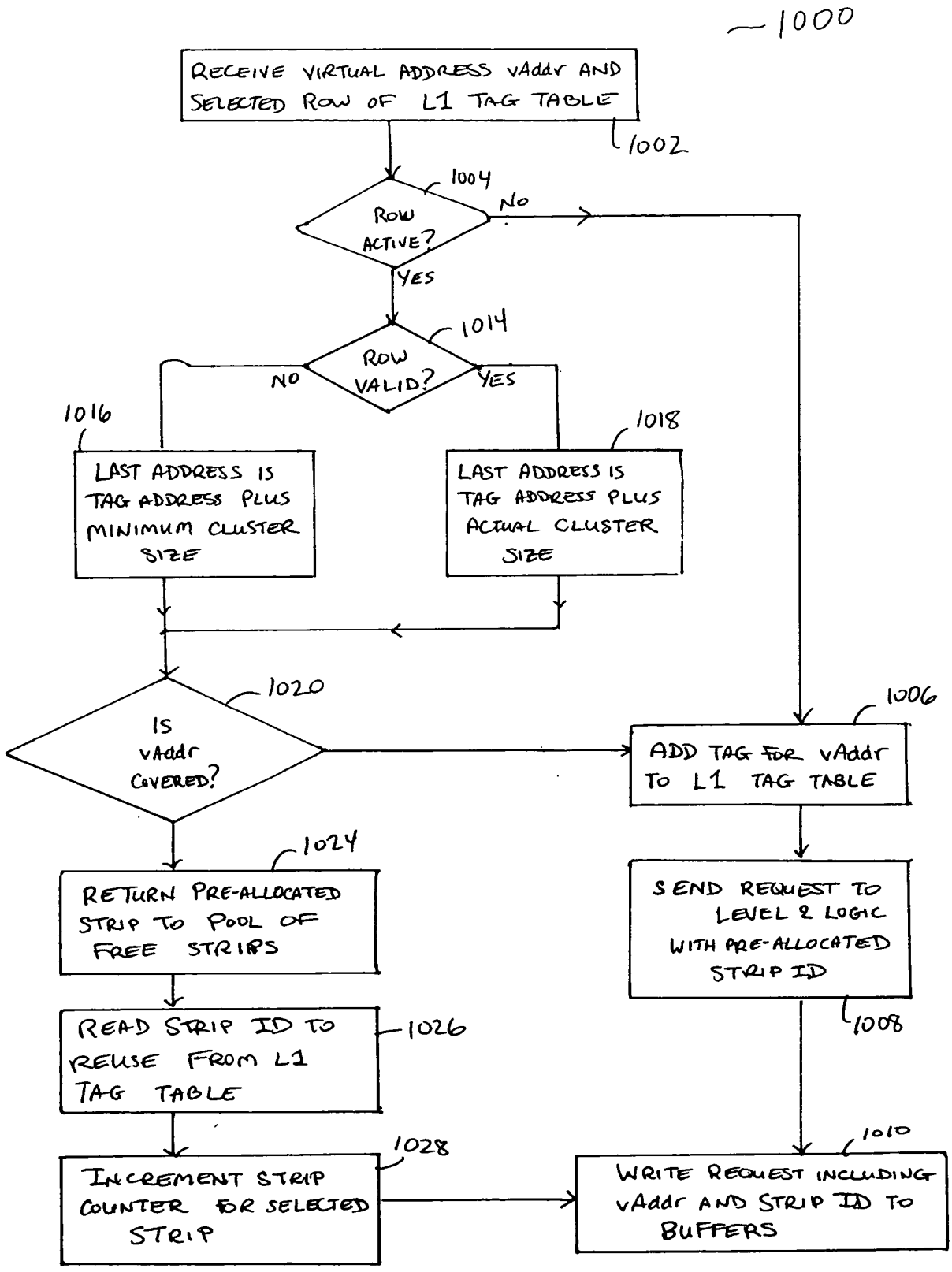
1000
RECEIVE VIRTUAL ADDRESS vAddr AND SELECTED ROW OF L1 TAG TABLE
1002
1004
ROW ACTIVE?
No
Yes
1014
ROW VALID?
NO
YES
1016
LAST ADDRESS IS TAG ADDRESS PLUS MINIMUM CLUSTER SIZE
1018
LAST ADDRESS IS TAG ADDRESS PLUS ACTUAL CLUSTER SIZE
1020
IS vAddr COVERED?
1006
ADD TAG FOR vAddr TO L1 TAG TABLE
1024
RETURN PRE-ALLOCATED STRIP TO POOL OF FREE STRIPS
SEND REQUEST TO LEVEL 2 LOGIC WITH PRE-ALLOCATED STRIP ID
1008
READ STRIP ID TO REUSE FROM L1 TAG TABLE
1026
1028
INCREMENT STRIP COUNTER FOR SELECTED STRIP
1010
WRITE REQUEST INCLUDING vAddr AND STRIP ID TO BUFFERS

FIG. 10

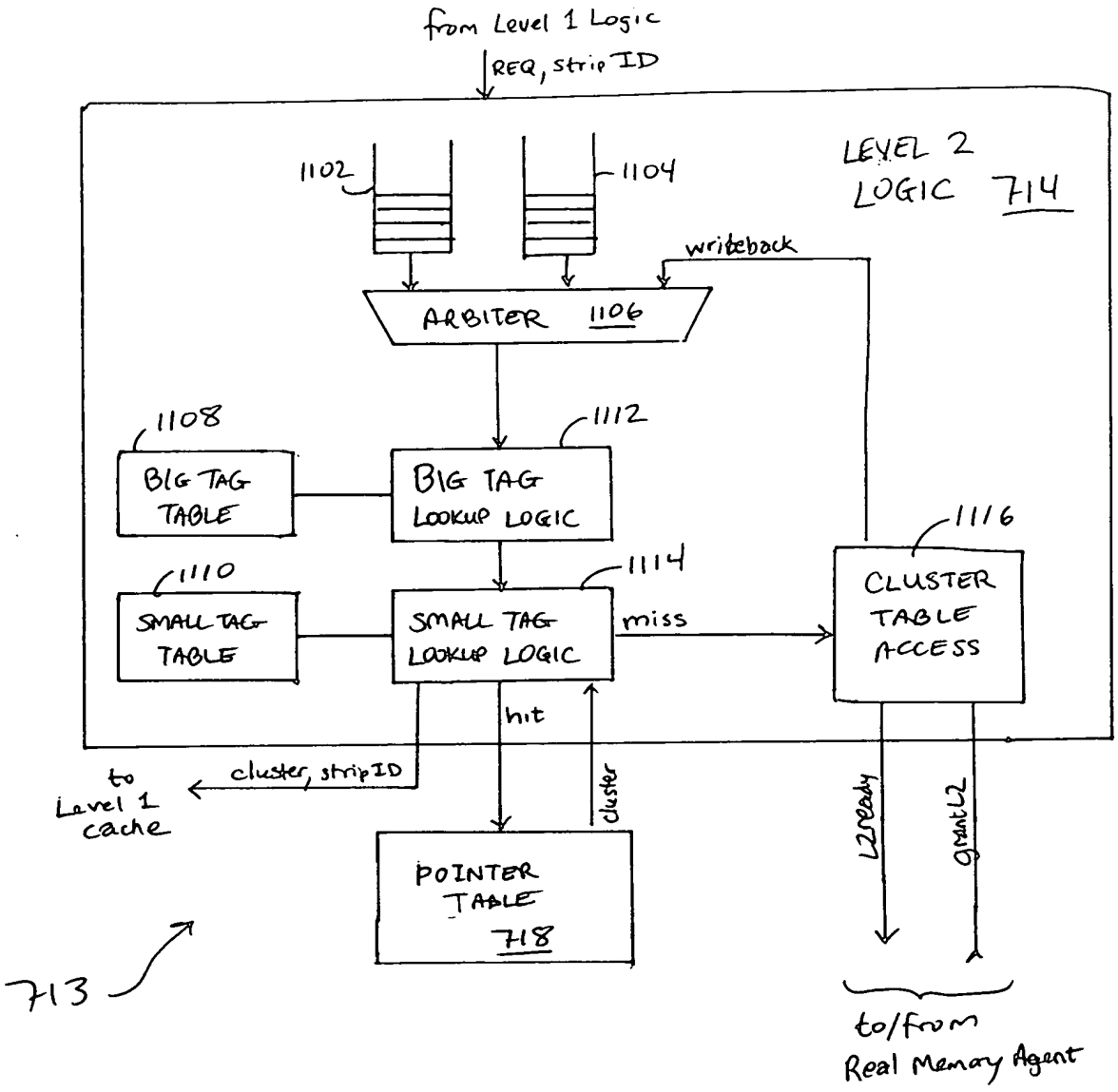
from Level 1 Logic
REQ, strip ID
LEVEL 2
LOGIC 714
1102
1104
writeback
ARBITER 1106
1108
BIG TAG
TABLE
1112
BIG TAG
LOOKUP LOGIC
1116
CLUSTER
TABLE
ACCESS
1110
SMALL TAG
TABLE
1114
SMALL TAG
LOOKUP LOGIC
miss
hit
cluster, strip ID
to
Level 1
cache
cluster
L2ready
grantL2
POINTER
TABLE
718
713
to/from
Real Memory Agent

FIG. 11

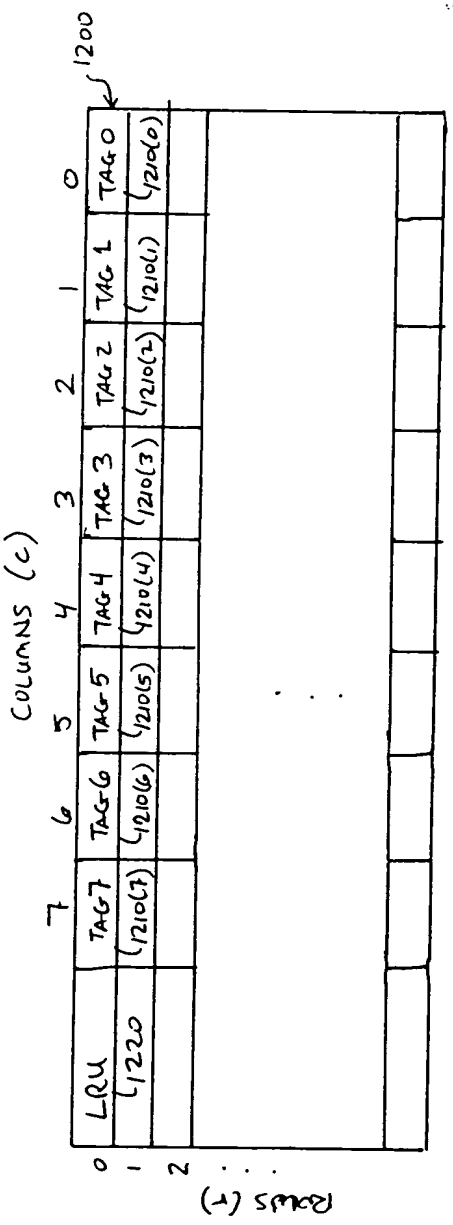
1200
COLUMNS (c)
0
1
2
3
4
5
6
7
TAG 0
TAG 1
TAG 2
TAG 3
TAG 4
TAG 5
TAG 6
TAG 7
1210(0)
1210(1)
1210(2)
1210(3)
1210(4)
1210(5)
1210(6)
1210(7)
LRU
1220
ROWS (r)
0
1
2

FIG. 12

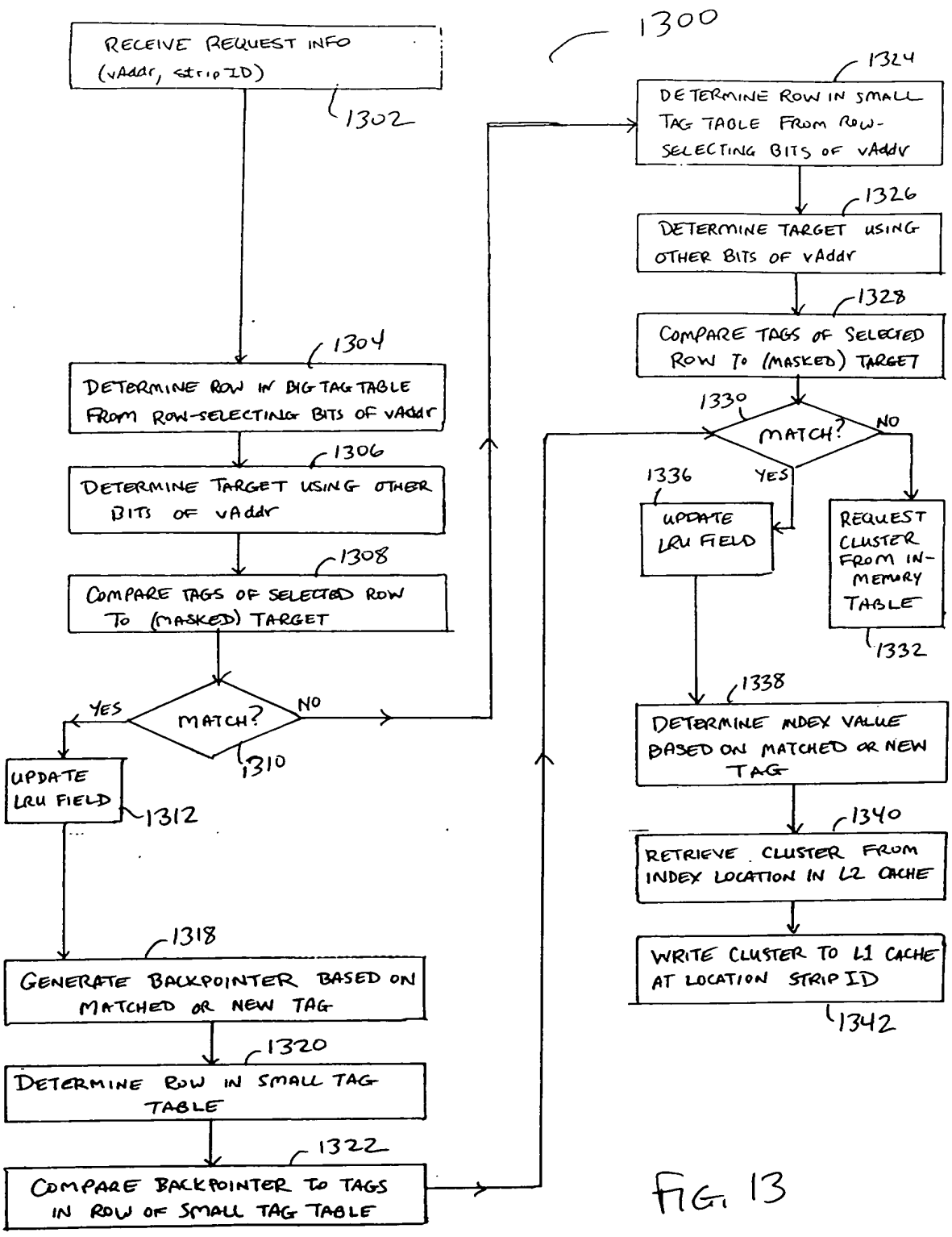
1300
RECEIVE REQUEST INFO (vAddr, strioID)
1302
DETERMINE ROW IN BIG TAG TABLE FROM ROW-SELECTING BITS OF vAddr
1304
DETERMINE TARGET USING OTHER BITS OF vAddr
1306
COMPARE TAGS OF SELECTED ROW TO (MASKED) TARGET
1308
MATCH?
YES
NO
1310
UPDATE LRU FIELD
1312
GENERATE BACKPOINTER BASED ON MATCHED OR NEW TAG
1318
DETERMINE ROW IN SMALL TAG TABLE
1320
COMPARE BACKPOINTER TO TAGS IN ROW OF SMALL TAG TABLE
1322
DETERMINE ROW IN SMALL TAG TABLE FROM ROW-SELECTING BITS OF vAddr
1324
DETERMINE TARGET USING OTHER BITS OF vAddr
1326
COMPARE TAGS OF SELECTED ROW TO (MASKED) TARGET
1328
MATCH?
YES
NO
1330
UPDATE LRU FIELD
1336
REQUEST CLUSTER FROM IN-MEMORY TABLE
1332
DETERMINE INDEX VALUE BASED ON MATCHED OR NEW TAG
1338
RETRIEVE CLUSTER FROM INDEX LOCATION IN L2 CACHE
1340
WRITE CLUSTER TO L1 CACHE AT LOCATION STRIP ID
1342

FIG. 13

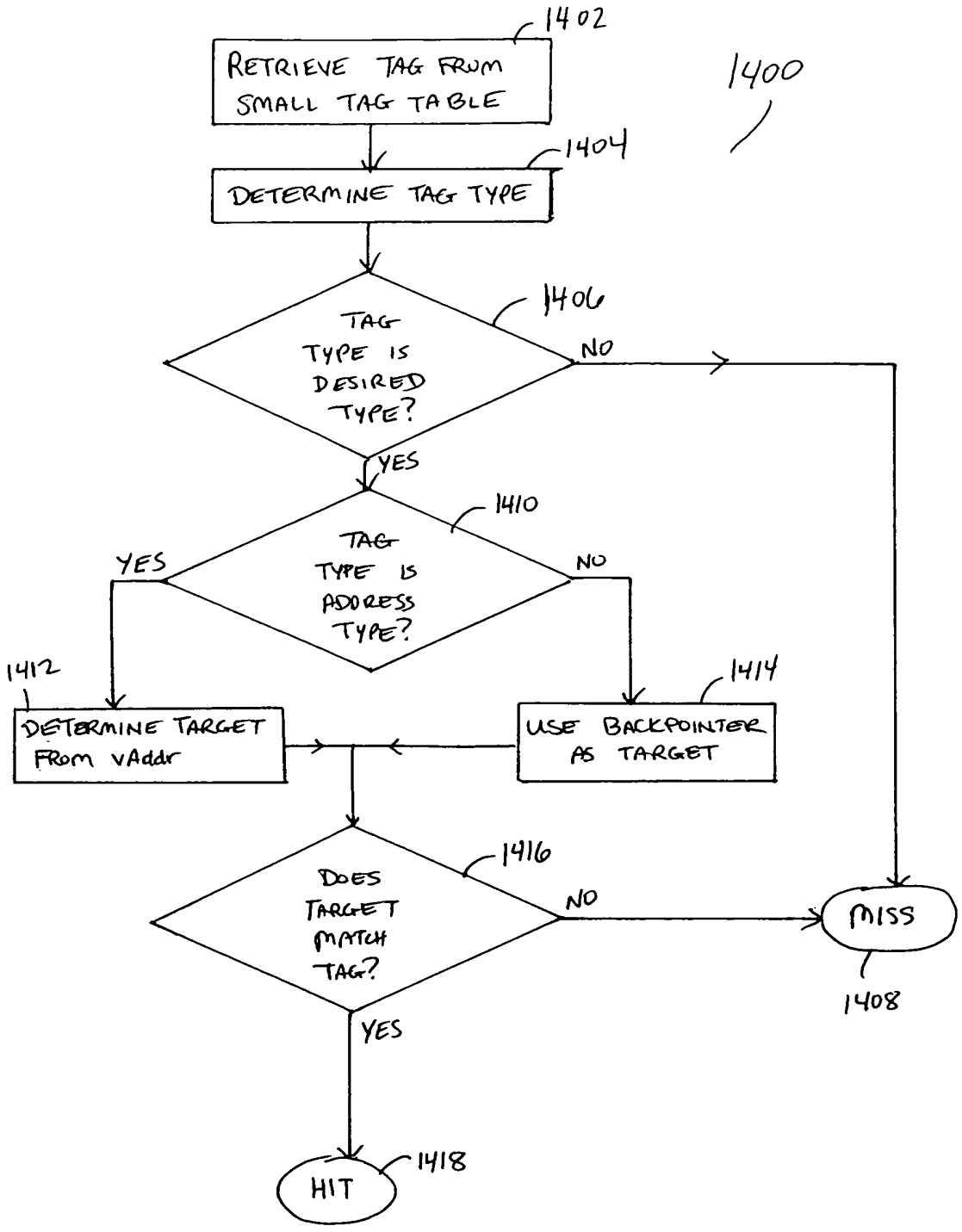
1400
1402
RETRIEVE TAG FROM SMALL TAG TABLE
1404
DETERMINE TAG TYPE
1406
TAG TYPE IS DESIRED TYPE?
NO
YES
1410
TAG TYPE IS ADDRESS TYPE?
YES
NO
1412
DETERMINE TARGET FROM vAddr
1414
USE BACKPOINTER AS TARGET
1416
DOES TARGET MATCH TAG?
NO
MISS
1408
YES
1418
HIT

FIG. 14

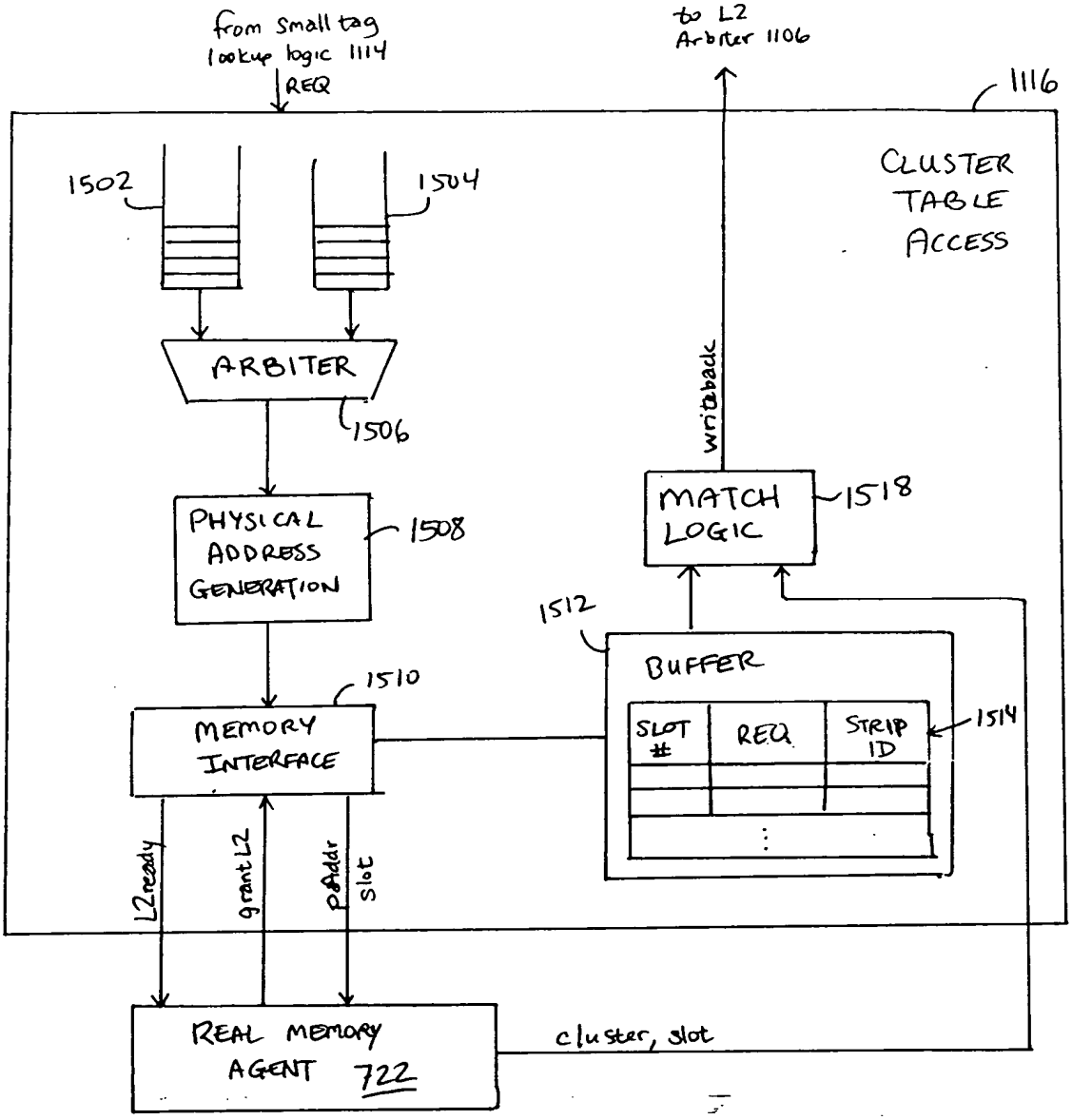
from Small tag lookup logic 1114
REQ
to L2 Arbiter 1106
1116
CLUSTER TABLE ACCESS
1502
1504
ARBITER
1506
writeback
MATCH LOGIC
1518
PHYSICAL ADDRESS GENERATION
1508
1512
BUFFER
1510
MEMORY INTERFACE
SLOT #
REQ
STRIP ID
1514
L2ready
grantL2
pAddr
slot
REAL MEMORY AGENT 722
cluster, slot

FIG. 15

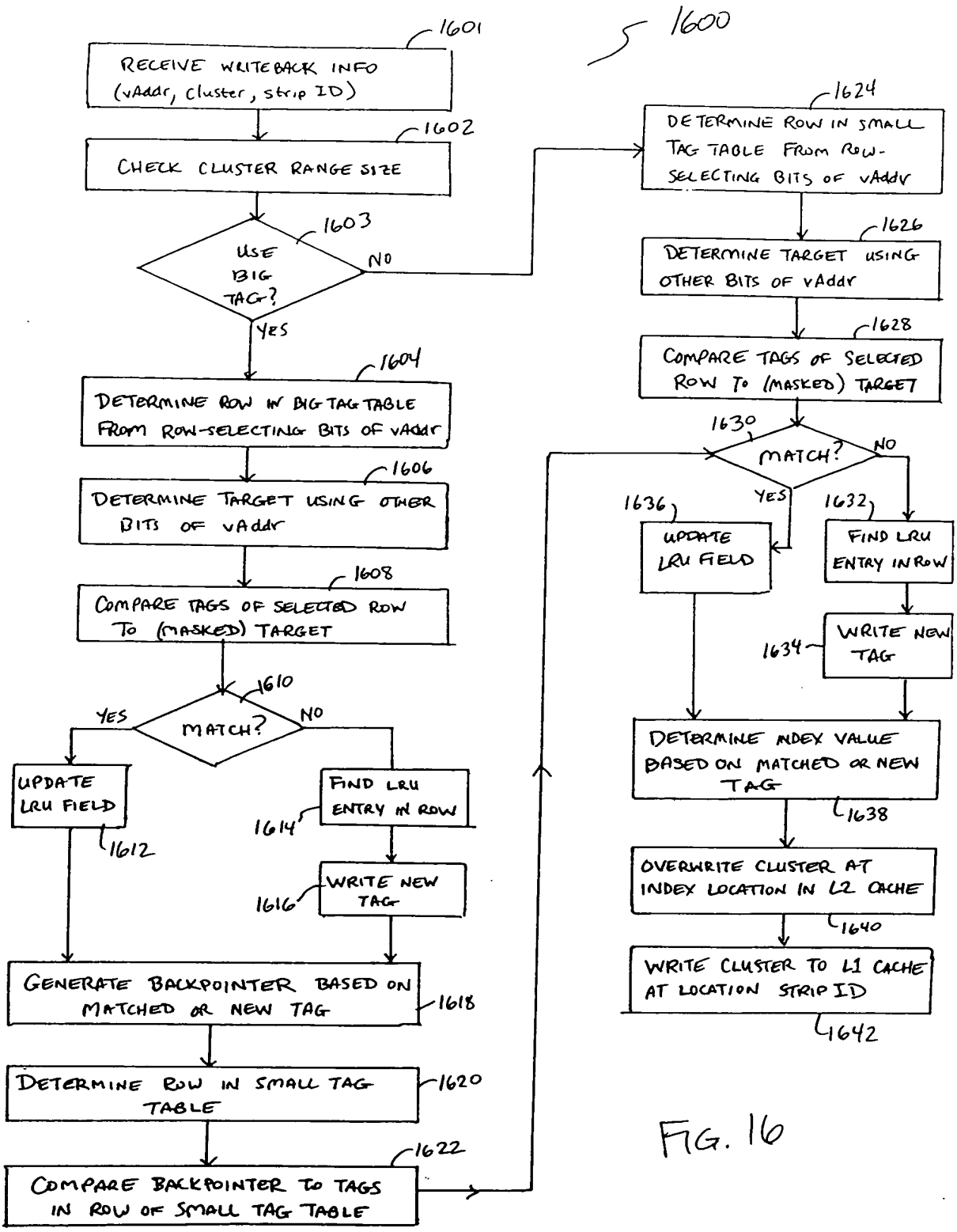
1600
1601
RECEIVE WRITEBACK INFO (vAddr, Cluster, Strip ID)
1602
CHECK CLUSTER RANGE SIZE
1603
USE BIG TAG?
NO
YES
1604
DETERMINE ROW IN BIG TAG TABLE FROM ROW-SELECTING BITS OF vAddr
1606
DETERMINE TARGET USING OTHER BITS OF vAddr
1608
COMPARE TAGS OF SELECTED ROW TO (MASKED) TARGET
1610
MATCH?
YES
NO
UPDATE LRU FIELD
1612
FIND LRU ENTRY IN ROW
1614
WRITE NEW TAG
1616
GENERATE BACKPOINTER BASED ON MATCHED OR NEW TAG
1618
DETERMINE ROW IN SMALL TAG TABLE
1620
COMPARE BACKPOINTER TO TAGS IN ROW OF SMALL TAG TABLE
1622
1624
DETERMINE ROW IN SMALL TAG TABLE FROM ROW-SELECTING BITS OF vAddr
1626
DETERMINE TARGET USING OTHER BITS OF vAddr
1628
COMPARE TAGS OF SELECTED ROW TO (MASKED) TARGET
1630
MATCH?
NO
YES
1636
UPDATE LRU FIELD
1632
FIND LRU ENTRY IN ROW
1634
WRITE NEW TAG
DETERMINE INDEX VALUE BASED ON MATCHED OR NEW TAG
1638
OVERWRITE CLUSTER AT INDEX LOCATION IN L2 CACHE
1640
WRITE CLUSTER TO L1 CACHE AT LOCATION STRIP ID
1642

FIG. 16

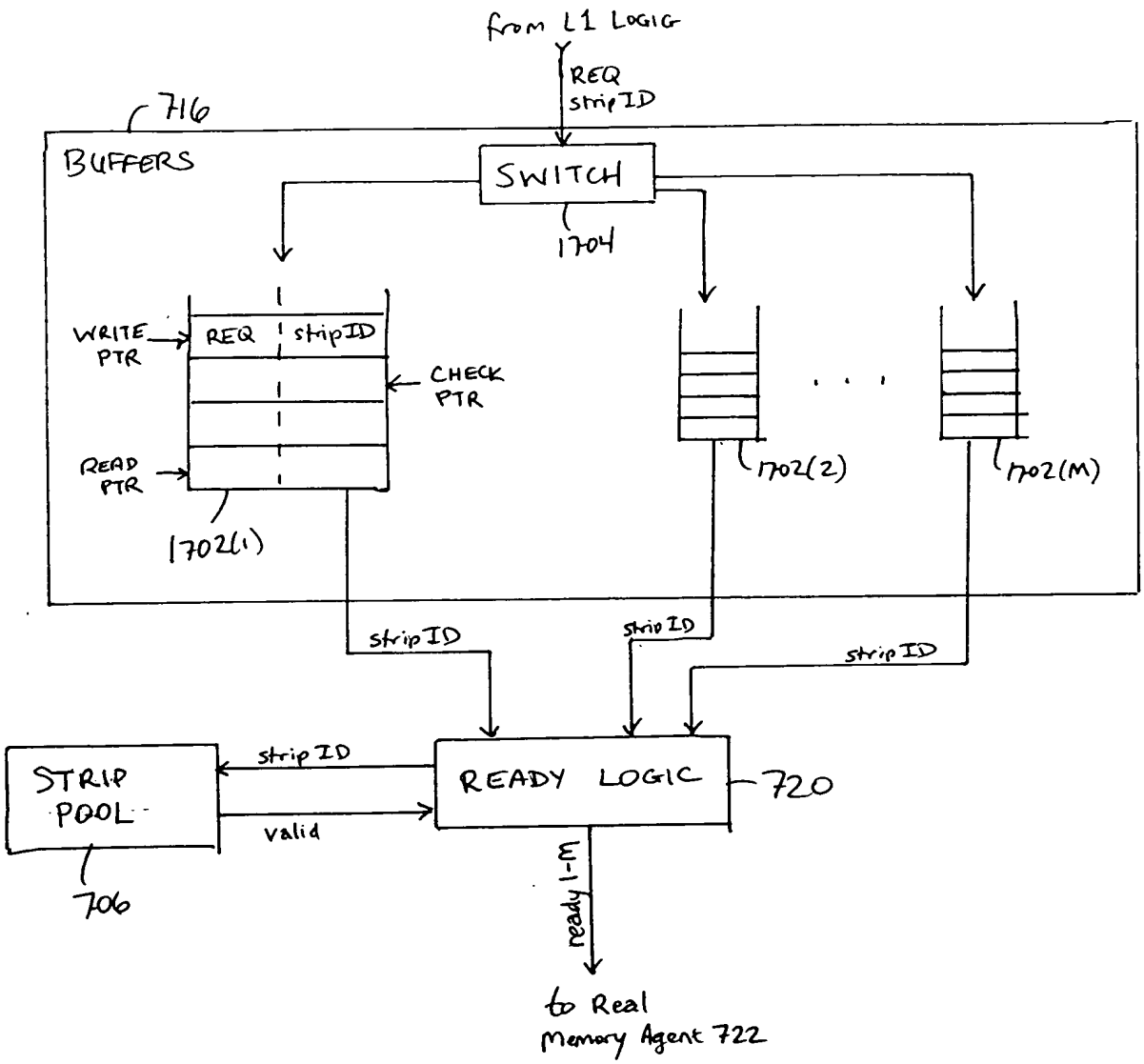
from L1 LOGIC
REQ
stripID
716
BUFFERS
SWITCH
1704
WRITE PTR
REQ
stripID
CHECK PTR
READ PTR
1702(1)
1702(2)
1702(M)
stripID
stripID
stripID
STRIP POOL
706
stripID
valid
READY LOGIC
720
ready 1-M
to Real
Memory Agent 722

FIG 17

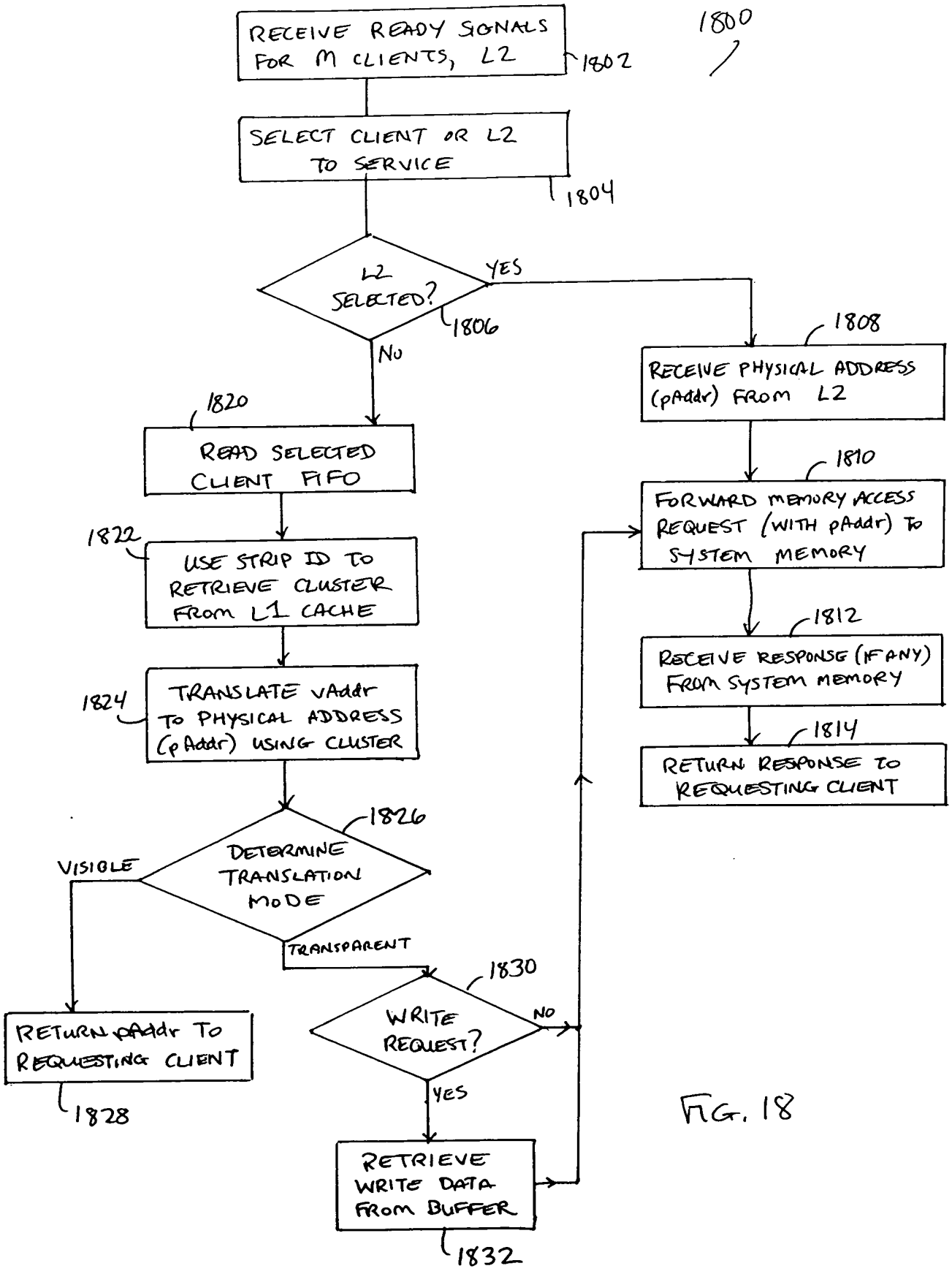
1800
RECEIVE READY SIGNALS FOR M CLIENTS, L2
1802
SELECT CLIENT OR L2 TO SERVICE
1804
L2 SELECTED?
1806
YES
NO
1808
RECEIVE PHYSICAL ADDRESS (pAddr) FROM L2
1810
FORWARD MEMORY ACCESS REQUEST (WITH pAddr) TO SYSTEM MEMORY
1812
RECEIVE RESPONSE (IF ANY) FROM SYSTEM MEMORY
1814
RETURN RESPONSE TO REQUESTING CLIENT
1820
READ SELECTED CLIENT FIFO
1822
USE STRIP ID TO RETRIEVE CLUSTER FROM L1 CACHE
1824
TRANSLATE vAddr TO PHYSICAL ADDRESS (pAddr) USING CLUSTER
1826
DETERMINE TRANSLATION MODE
VISIBLE
TRANSPARENT
RETURN pAddr TO REQUESTING CLIENT
1828
1830
WRITE REQUEST?
NO
YES
RETRIEVE WRITE DATA FROM BUFFER
1832

FIG. 18

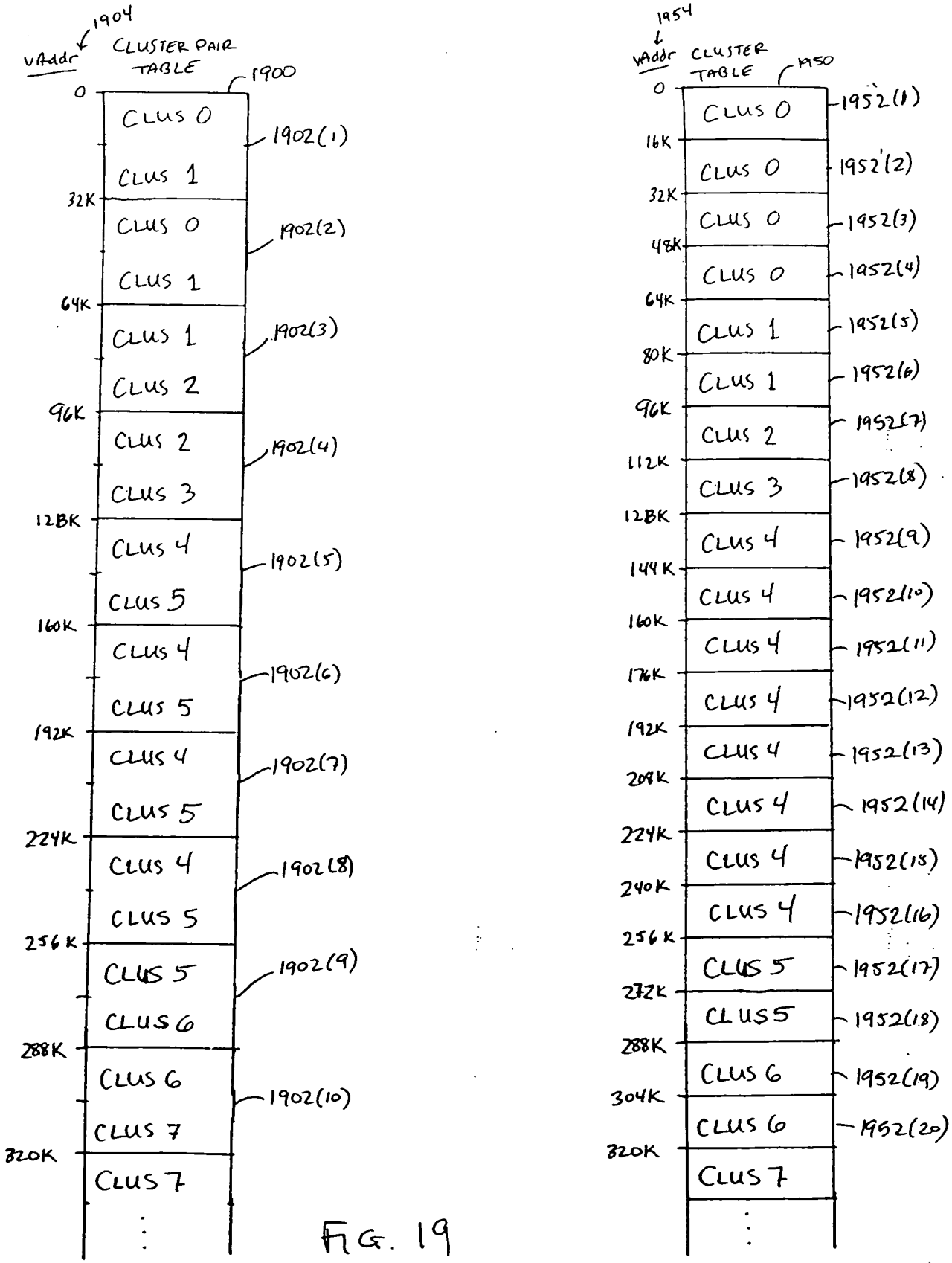
1904
vAddr
CLUSTER PAIR TABLE
1900
0
CLUS 0
1902(1)
CLUS 1
32K
CLUS 0
1902(2)
CLUS 1
64K
CLUS 1
1902(3)
CLUS 2
96K
CLUS 2
1902(4)
CLUS 3
128K
CLUS 4
1902(5)
CLUS 5
160K
CLUS 4
1902(6)
CLUS 5
192K
CLUS 4
1902(7)
CLUS 5
224K
CLUS 4
1902(8)
CLUS 5
256K
CLUS 5
1902(9)
CLUS 6
288K
CLUS 6
1902(10)
CLUS 7
320K
CLUS 7
1954
vAddr
CLUSTER TABLE
1950
0
CLUS 0
1952(1)
16K
CLUS 0
1952(2)
32K
CLUS 0
1952(3)
48K
CLUS 0
1952(4)
64K
CLUS 1
1952(5)
80K
CLUS 1
1952(6)
96K
CLUS 2
1952(7)
112K
CLUS 3
1952(8)
128K
CLUS 4
1952(9)
144K
CLUS 4
1952(10)
160K
CLUS 4
1952(11)
176K
CLUS 4
1952(12)
192K
CLUS 4
1952(13)
208K
CLUS 4
1952(14)
224K
CLUS 4
1952(15)
240K
CLUS 4
1952(16)
256K
CLUS 5
1952(17)
272K
CLUS 5
1952(18)
288K
CLUS 6
1952(19)
304K
CLUS 6
1952(20)
320K
CLUS 7

FIG. 19

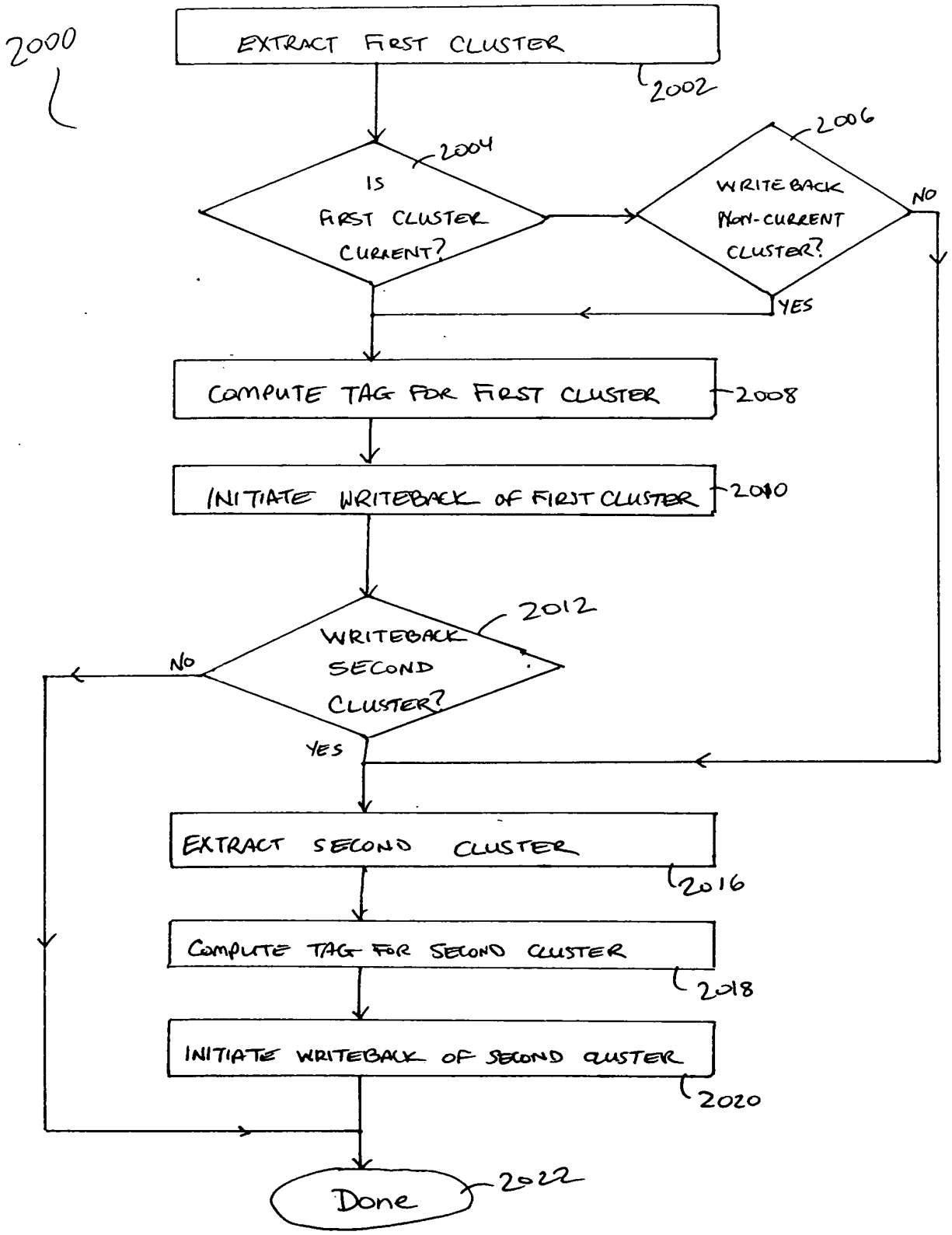
2000
EXTRACT FIRST CLUSTER
2002
2004
IS FIRST CLUSTER CURRENT?
2006
WRITE BACK NON-CURRENT CLUSTER?
NO
YES
COMPUTE TAG FOR FIRST CLUSTER
2008
INITIATE WRITEBACK OF FIRST CLUSTER
2010
2012
WRITEBACK SECOND CLUSTER?
No
YES
EXTRACT SECOND CLUSTER
2016
COMPUTE TAG FOR SECOND CLUSTER
2018
INITIATE WRITEBACK OF SECOND CLUSTER
2020
Done
2022

FIG. 20

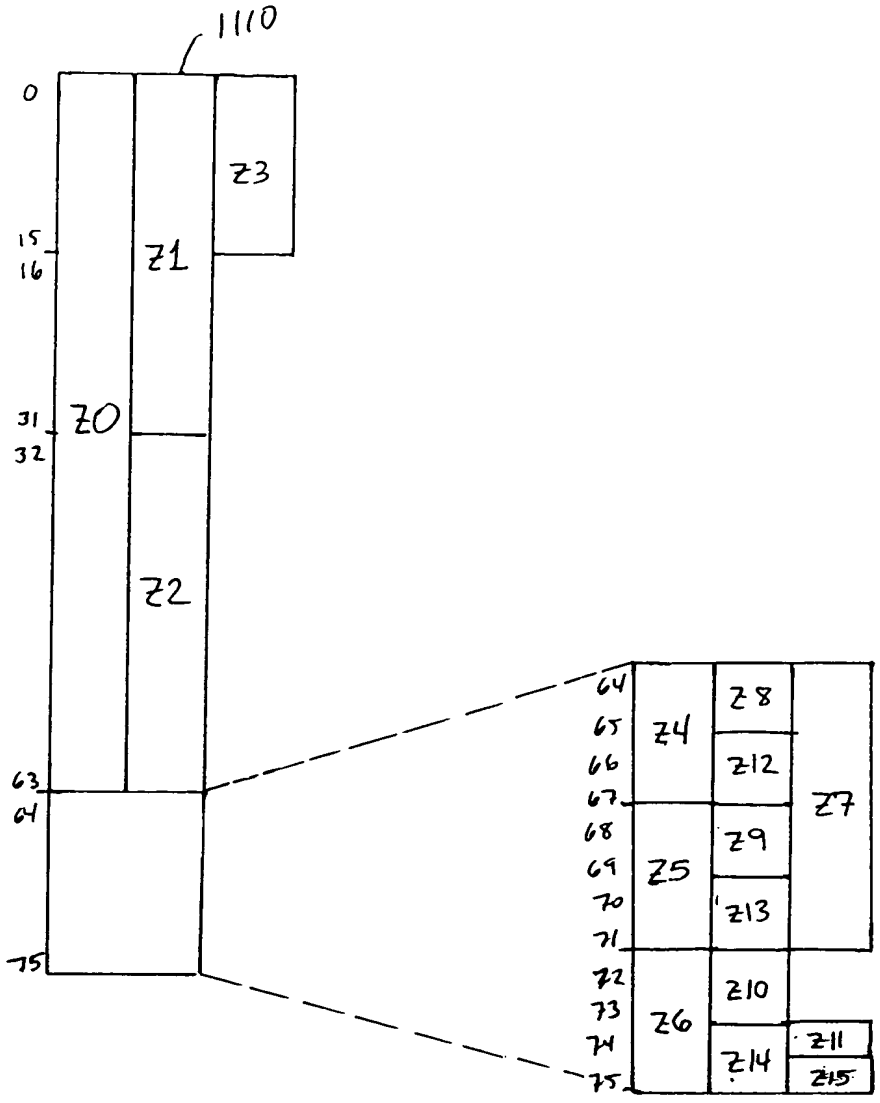
1110
0
15
16
31
32
63
64
75
Z0
Z1
Z2
Z3
64
65
66
67
68
69
70
71
72
73
74
75
Z4
Z5
Z6
Z7
Z8
Z12
Z9
Z13
Z10
Z14
Z11
Z15

FIG. 21

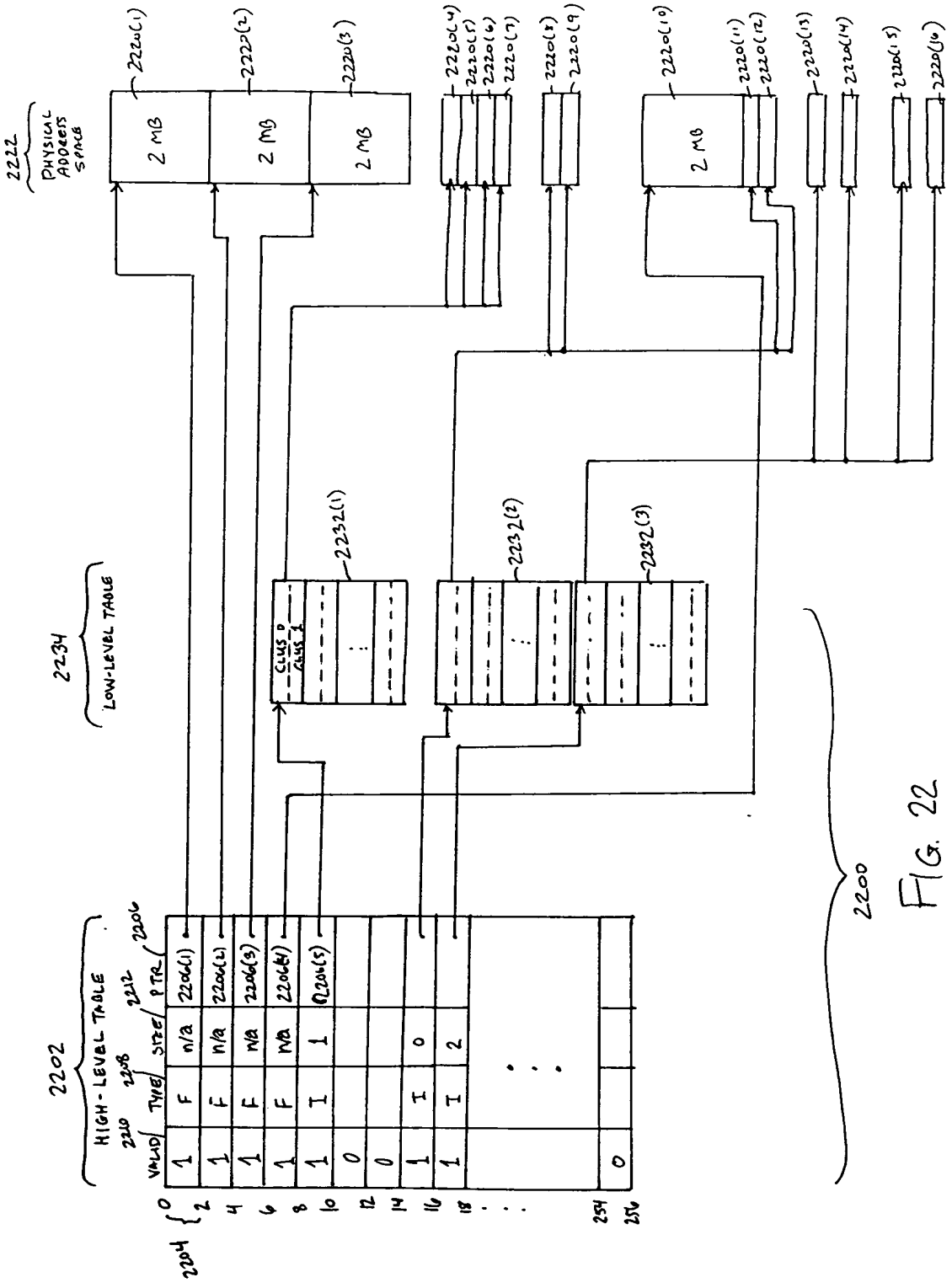
2222
PHYSICAL ADDRESS SPACE
2220(1)
2220(2)
2220(3)
2220(4)
2220(5)
2220(6)
2220(7)
2220(8)
2220(9)
2220(10)
2220(11)
2220(12)
2220(13)
2220(14)
2220(15)
2220(16)
2 MB
2234
LOW-LEVEL TABLE
2232(1)
2232(2)
2232(3)
CLUS D
CLUS 1
2202
HIGH-LEVEL TABLE
210
208
212
206
VALID
TYPE
SIZE
PTR
2206(1)
2206(2)
2206(3)
2206(4)
2206(5)
n/a
N/A
1
0
2
F
I
0
2
4
6
8
10
12
14
16
18
254
256
2204
2200

FIG. 22

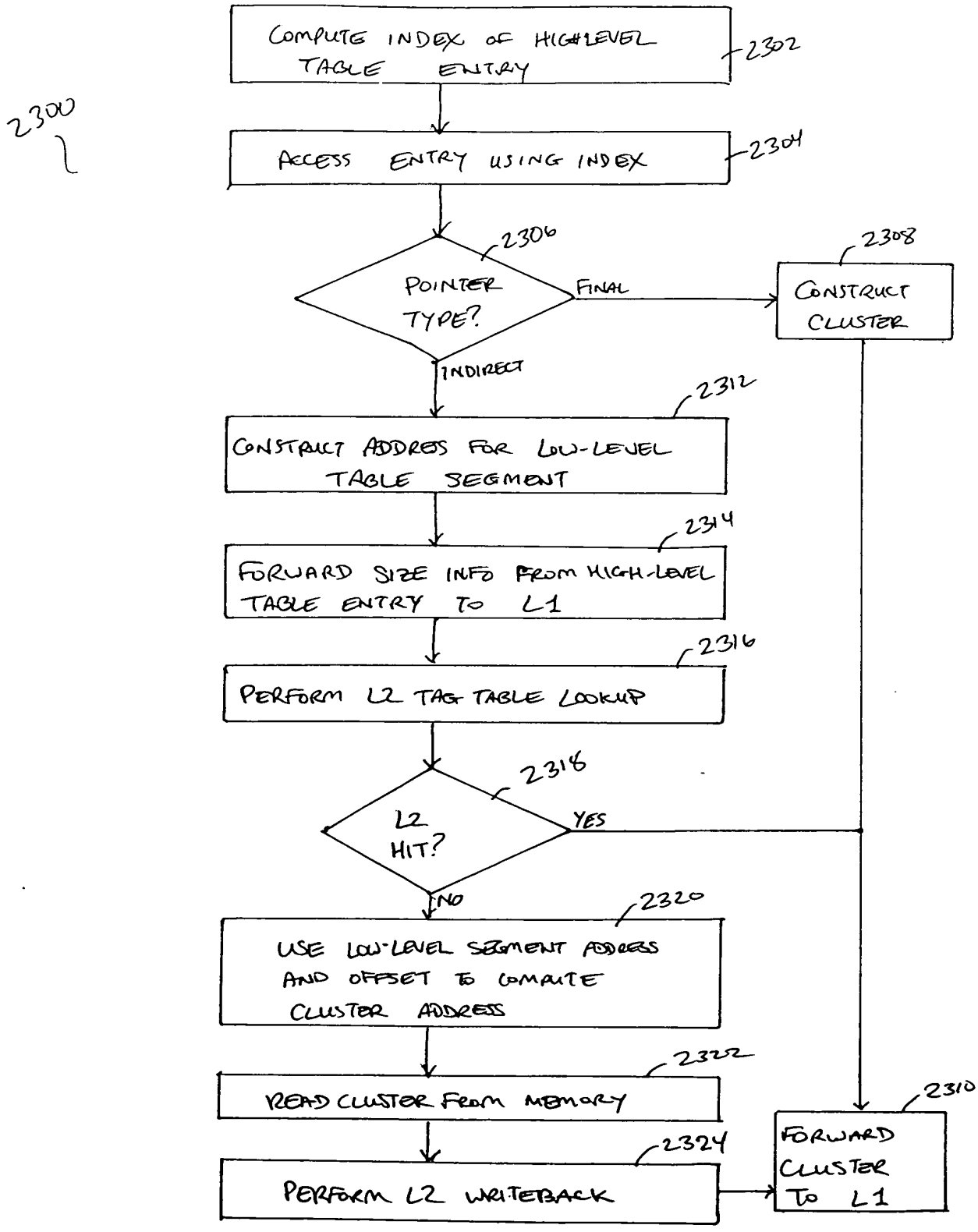
2300
COMPUTE INDEX OF HIGH-LEVEL TABLE ENTRY
2302
ACCESS ENTRY USING INDEX
2304
POINTER TYPE?
2306
FINAL
CONSTRUCT CLUSTER
2308
INDIRECT
CONSTRUCT ADDRESS FOR LOW-LEVEL TABLE SEGMENT
2312
FORWARD SIZE INFO FROM HIGH-LEVEL TABLE ENTRY TO L1
2314
PERFORM L2 TAG TABLE LOOKUP
2316
L2 HIT?
2318
YES
NO
USE LOW-LEVEL SEGMENT ADDRESS AND OFFSET TO COMPUTE CLUSTER ADDRESS
2320
READ CLUSTER FROM MEMORY
2322
PERFORM L2 WRITEBACK
2324
FORWARD CLUSTER TO L1
2310

FIG. 23

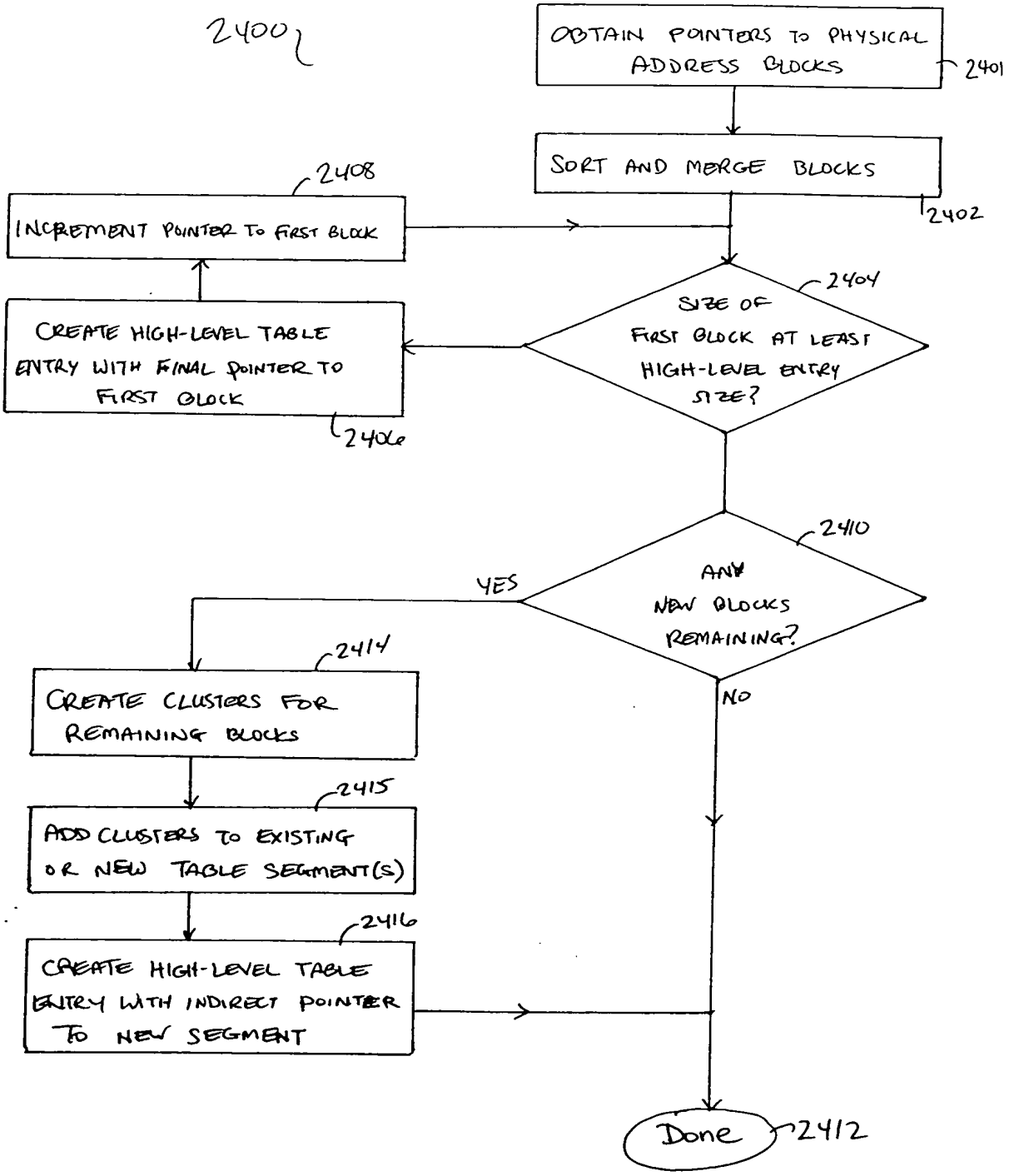
2400
OBTAIN POINTERS TO PHYSICAL ADDRESS BLOCKS
2401
SORT AND MERGE BLOCKS
2402
2404
SIZE OF FIRST BLOCK AT LEAST HIGH-LEVEL ENTRY SIZE?
2406
CREATE HIGH-LEVEL TABLE ENTRY WITH FINAL POINTER TO FIRST BLOCK
2408
INCREMENT POINTER TO FIRST BLOCK
2410
ANY NEW BLOCKS REMAINING?
YES
NO
2414
CREATE CLUSTERS FOR REMAINING BLOCKS
2415
ADD CLUSTERS TO EXISTING OR NEW TABLE SEGMENT(S)
2416
CREATE HIGH-LEVEL TABLE ENTRY WITH INDIRECT POINTER TO NEW SEGMENT
Done
2412

FIG. 24